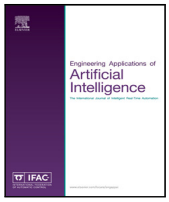
Engineering Applications of Artificial Intelligence 123 (2023) 106343

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Real-time propeller fault detection for multirotor drones based on vibration data analysis

Alessandro Baldini, Riccardo Felicetti *, Francesco Ferracuti, Alessandro Freddi, Sabrina Iarlori, Andrea Monteriù

Department of Information Engineering, Università Politecnica delle Marche, Via Brecce Bianche 12, Ancona, 60131, Italy

ARTICLE INFO

Keywords:
Unmanned aerial vehicles
Fault detection
Signal processing

ABSTRACT

This article presents a Fault Detection (FD) method to deal with propeller faults on multirotor drones in real-time. Several solutions have been proposed in the literature, however, they depend on additional sensors and/or dedicated hardware to deal with heavy computational complexity. So, they cannot be implemented in off-the-shelf commercial devices, i.e., without the aid of additional on-board sensors and/or extra computational power. The proposed method, instead, requires the on-board Inertial Measurement Unit (IMU) data only: by combining Finite Impulse Response (FIR), together with sparse classifiers, only a subset of the features is actually needed online and the FD is thus feasible in real-time.

Design and tests are based on real flight data from a hexarotor, equipped with a conventional ArduPilot-based controller. The classification accuracy in testing is up to 93.37% (98.21%) with a binary tree (Linear Support Vector Machine (LSVM)). Moreover, the space and time complexity of the proposed method is low: on a PixHawk Cube flight controller, it requires less than 2% of the cycle time, and can then run in real-time. Finally, the proposed fault detection solution is model-free and it can be easily generalized to other multirotor vehicles.

1. Introduction

Unmanned Aerial Vehicles (UAVs) represent a revolution for civilian and military purposes because they provide a cost-effective solution in a variety of applications, such as last-mile delivery, photography, search and rescue, infrastructure monitoring, patrolling, intelligence activities, and aerial warfare. As UAVs are pervading commercial and private sectors, including many nonprofessional users for recreational purposes, the main concerns of people are privacy and security (Tan et al., 2021). While privacy protection is primarily up to lawmakers and regulatory agencies, engineers can play a central role in improving security. In fact, small-scale UAVs are characterized by a high loss rate (Shraim et al., 2018) and faults are crucial: according to Wild et al. (2016), 64% of accidents and incidents involving remotely piloted UAV are caused by equipment problems. In particular, actuator faults can have severe consequences if not tackled in time, as they can cause damages and loss of control in flight, and Fault Detection (FD) is the first step to mitigate the effects of faults.

The weight of the on-board sensors and payloads represents a limitation, especially in the case of small-size UAVs, as every additional equipment diminishes both autonomy and maneuverability. Also, the cost of the on-board instrumentation should be taken into account, as it could easily exceed the cost of the UAV itself (Wang et al., 2020b). In the literature (see Section 2), several strategies have been proposed, but they depend on (i) additional sensors and/or (ii) dedicated hardware to deal with heavy computational complexity. So, they cannot be implemented in off-the-shelf commercial devices, i.e., without the aid of additional on-board sensors and/or extra computational power. Hence, in this paper, we propose a FD strategy to tackle propeller faults on multirotor UAVs whose main feature is the ease of implementation in commercial devices with limited computational and sensing capabilities: the number of FD features is small and the computation of both the features and the decision is straightforward. As a consequence, there is no need for additional equipment, such as additional sensors or extra computational power: only the on-board acceleration measurements from the built-in Inertial Measurement Unit (IMU) are employed for FD. Although we investigate frequency-domain features, we show that they can be extracted by means of a few Finite Impulse Response (FIR) filters, without the need for performing Fast Fourier Transform (FFT) online. Essentially, the main novelty of this work is the introduction of an Artificial Intelligence (AI)-based FD algorithm to detect propeller faults that can run online in commercial devices, without the need for additional payloads or computational power. The proposed strategy is also novel because neither predefined maneuvers nor intensive UAV

* Corresponding author.
E-mail address: r.felicetti@staff.univpm.it (R. Felicetti).

https://doi.org/10.1016/j.engappai.2023.106343
Received 10 January 2023; Received in revised form 30 March 2023; Accepted 14 April 2023
Available online 3 May 2023

modeling are needed: in fact, we show that the proposed method can be applied successfully in real, manual flight mode conditions.

The article is structured as follows. In Section 2, we review the state of the art for the FD of actuator faults for UAVs. In Section 3, we introduce the experimental setup, including the hardware specifications, the customization of the ArduPilot firmware to enhance data acquisition, the software setup, and finally the acquisition of experimental data with different propeller faults. The experimental data is described in Section 4, where data preprocessing is followed by a detailed statistical analysis to find out solid features for FD. In Section 5, we design two AI-based FD strategies, i.e., the proposed one that exploits FIR filtering-based features, together with lightweight classifiers, and, in contrast, the same kind of classifiers while making use of a collection of conventional features for FD. The results are then compared in Section 6, where the accuracy, the execution time during Hardware-In-the-Loop (HIL) testing, and the testing with an additional faulty blade are discussed. Conclusions and future works end the article in the last Section.

2. State of the art

Two main approaches are adopted in the literature to perform FD for UAVs: model-based and signal-based. As for the model-based approaches, methods such as Thau observers (Freddi et al., 2012), linear and Linear Parameter Varying (LPV) proportional–integral observers (Ortiz-Torres et al., 2020), and banks of nonlinear disturbance observers (Baldini et al., 2022) are well established in the scientific literature. Model-based strategies can go beyond FD: fault isolation and fault estimation are feasible, under specific assumptions, for many classes of faults. However, model-based approaches require a significant modeling effort and they clash with the unavailability of many parameters in the applications (e.g., inertia, lift and drag coefficients, friction). Indeed, using model-based approaches, the loss of effectiveness that may be perceived with a blade chipping is completely masked by variability in the performance of the brushed motors (Ghalamchi et al., 2019). Signal-based strategies, instead, do not require complex system models or assumptions on the fault. Provided that a suitable amount of data is available, including all of the possible faults of practical interest, an AI-based FD system can be defined, which consists of signal-based methods for feature generation and a classification tool (Gangsar and Tiwari, 2020). Many AI-based FD strategies for UAVs have been investigated in the last years: Ai et al. (2022) proposed a random forest to detect sensor faults, Liang et al. (2022) employed several variants of Kernel Principal Component Analysis (KPCA) to detect actuator and sensor faults, while Park et al. (2022) suggested to use deep neural networks to detect and isolate an actuator failure.

However, among the many FD strategies for UAVs (see, for example, Fourlas and Karras (2021) for a detailed analysis), vibration signals are not commonly employed as a diagnostic feature. This is somewhat surprising, as almost any UAV has built-in accelerometers, embedded in its flight controller, so the vibration data is available without the need for additional sensors. In the last years, the use of acceleration signals to check the health status of rotating machinery (Peng et al., 2005; Gangsar and Tiwari, 2020), as well as machine elements, such as bearings (Hoang and Kang, 2019), clutches (Chakrapani and Sugumaran, 2023), and milling cutters (Wu and Lei, 2019), has been broadly investigated in the literature, together with AI-based FD using electric measurements (Boztas and Tuncer, 2022; Thomas et al., 2023; Wang et al., 2023). Clearly, in the case of UAVs, the accelerometers are installed on the frame, and not on the single actuator as in large, industrial rotating machinery: this makes FD more challenging, because the vibration generated by several actuators, as well as sensor noise and disturbances (such as aerodynamic forces due to maneuvering), merge into the same acceleration measurements. In the literature, few FD strategies based on acceleration signals have been proposed for UAVs. The authors of Zhang et al. (2021) performed a wavelet packet decomposition on a quadrotor vibration data to extract FD features, and the decision was taken by a Long and Short-Term Memory (LSTM) artificial neural network. The computational effort is relevant and so it must be performed on a companion computer. In the previous work (Benini et al., 2019), the authors proposed a FD algorithm, based on acceleration signals, for a fixed-wing UAV. As the authors pointed out, the approach worked best in the take-off phase, and it required dedicated hardware to be run online due to computational issues. In Ghalamchi and Mueller (2018), the authors investigated the use of acceleration measurements to detect propeller faults; however, according to the authors, their approach could be employed when the quadrotor follows a predefined trajectory. Moreover, the approach relies on performing the FFT, which is impractical in commercial flight controllers. It also requires the buffering of a sequence of accelerometer measurements, so the data can be analyzed offline only. In Ghalamchi et al. (2019), instead, the authors propose to estimate the unbalance due to a chipped blade using an Extended Kalman Filter (EKF). The authors show that fault isolation is feasible, however, for a medium-sized hexarotor, the convergence time is in the order of minutes, which could be unsatisfactory for safety purposes.

In most of the literature, only dedicated hardware is considered to develop and test the FD solutions, whereas, in this work, the FD solution is deployed on a flight controller board with limited computational resources. In Bronz et al. (2020), the authors proposed a real-time FD solution tested on a dedicated board. Sensory information is passed to the inference computer (RaspberryPi-Zero) at a frequency of 10 Hz, and piling up the buffer (20 samples and 8 features) takes 2 s. The inference prediction is called only once the feature trajectory buffer is completed, so every 2 s and the prediction took about 0.06 s of computation time. The authors suggested that using an even more powerful embedded board such as Jetson Nano/Xavier-NX could increase the prediction frequency by at least one order of magnitude. Keipour et al. (2019) proposed a real-time approach using the recursive least squares method to detect anomalies in the behavior of an aircraft. They implemented it in Linux Ubuntu 16.04 (Xenial) using C++11 language and Robot Operating System (ROS) Kinetic Kame. An Nvidia Jetson TX2 was added to the base platform to deploy the FD solution. In the work of Simlinger and Ducard (2019), sensor faults did not occur during the experiments but were introduced artificially to the recorded experimental data, which was processed in real-time on a dedicated platform such as Intel Atom processor. Khalastchi et al. (2011), Zhang et al. (2020), Sun et al. (2017) proposed online/real-time FD solutions but their solution has not been implemented and deployed in the real hardware and then tested only in flight simulators, such as MATLAB/Simulink flight simulation platform. A HIL simulation is considered to test the FD solution by Xian and Hao (2019). The solution is implemented by using xPC target. A compact PC/104 computer is utilized as the target computer, and a laptop PC is employed as the host computer. The desktop PC was used as a flight visualization computer which runs FlightGear and Google Earth to show visual data, such as the orientation and flight path of the quadrotor UAV. In the study of Kantue and Pedro (2020), for the real-time testing of FD algorithm, the HIL simulation is achieved by a dedicated embedded board for the purpose of FD only, i.e., 180 MHz ARM Cortex-M4 with Floating Point Unit (Teensy 3.6) and 256KB RAM. Chen et al. (2017) proposed a FD algorithm, tested on a high-performance board, i.e., a Xilinx Zynq-7000 SoC with ARM processor. Based on the Flight Gear v3.4 flight simulator, a self-guided Cessna 172P aircraft was used to simulate the complete autonomous flight. The data generated by the flight simulator were sampled to 5 Hz. In the simulation, a kind of anomaly is injected into the vertical velocity indicator, so the vertical speed reading is abnormal in the flight data. The time required to process 4000 data points in the high-performance board is 48 ms. A Xilinx Zynq 7045-based airborne embedded computing platform in a real fixed-wing UAV platform is used to deploy the FD algorithm by Wang et al. (2020a). Zynq is a commonly used system on a chip that contains dual-core

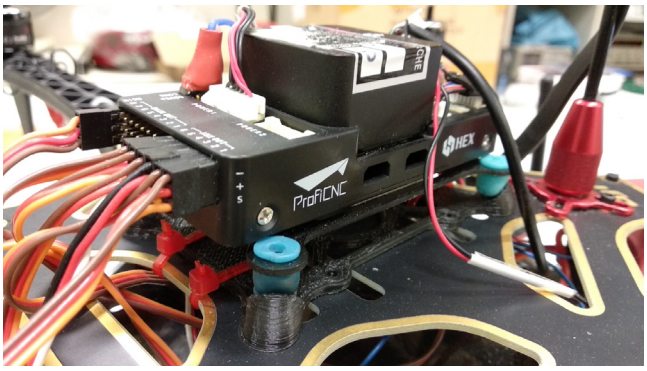


Fig. 1. Detail of the anti-vibration platform.

ARM (cortex-A9 arm) and FPGA. In this study, FPGA was dedicated for implementing the FD solution. Thus, to the best of the authors' knowledge, no FD solutions are available in the literature to deal with faults (more specifically, actuator faults) that are proven to be feasible in real-time on a conventional flight controller, without the need for external sensors and/or dedicated computing boards.

3. Experimental setup

Real flight data from a DJI F550 hexarotor under actuator faults were acquired to design and validate the proposed FD algorithm. In the following, we detail the hardware equipment and the software setup, including firmware customization to enhance data logging. The objective is to create a dataset that includes flights in different healthy and faulty conditions: for this purpose, data are logged on a Secure Digital (SD) card so that they are available for offline analysis.

3.1. Hardware specifications

The hexarotor has been internally assembled, starting from the following commercial components.

- Flight controller: PixHawk PX4 Cube Black
- IMU: MPU9250 (InvenSense, 2016)
- Frame: DJI FlameWheel F550
- Battery: Tattu 25C LiPo, 4 cells, 6700 mAh
- Outrunner motors: T-Motor Air Gear 350
- Self-tightening propellers: T-Motor T9545
- Electronic Speed Controller (ESC): Tekko Holybro D-Shot 125
- Remote control: FrSky Taranis X8R (receiver) and X9D (transmitter)

Please note that the flight controller is installed on a commercial anti-vibration platform with four rubber dampeners (see Fig. 1), which is a widespread expedient to reduce the vibrations measured by the IMU, thus enhancing the flight control performances. Despite the dampening, in the remainder, we show that high-frequency acceleration measurements using the on-board IMU are still possible and meaningful for FD purposes.

3.2. Custom firmware and software setup

When enabled, the default ArduPilot data log rate for IMU data on the SD card is 25 Hz, which is too low to capture high-frequency dynamics. The default frequency is hardcoded: in order to increase the log rate, the logging frequency must be manually increased in the ArduPilot source code. So, first of all, the latest (V4.3.1) ArduPilot version was downloaded from the official GitHub repository.[1] To modify the data log frequency, the "Copter.cpp" file was modified in Eclipse.[2] In particular, the scheduled frequency of the task "twentyfive_hz_logging" was increased from 25 Hz to 350 Hz.

The target logging frequency, i.e., 350 Hz, was the result of a trial-and-error procedure. Clearly, the higher the frequency, the better for diagnostic purposes. On the other hand, higher rates compromise the execution of the remaining tasks and thus they can alter the behavior in flight. In fact, the ArduPilot code is organized into tasks: the scheduler runs at 400 Hz, while each task is scheduled with a predefined rate (less or equal to 400 Hz). The scheduler executes the main tasks at 400 Hz, then the secondary tasks are executed in the remaining time (i.e., less than 2.5 ms), if needed (according to the desired task rate). Given a predefined (hardcoded) estimation of the task duration, if the time left in the current iteration is not sufficient to perform a given secondary task, that secondary task is postponed to the next iteration. When the target logging frequency is too large, the first symptom is that the logging rate is not constant. This means that the scheduler cannot perform the tasks at the prescribed rate, and the secondary tasks with lower priority are affected as well.

The source code was compiled using Cygwin.[3] and a suitable GCC compiler[4] In order to load the custom firmware on the hexarotor and configure it, the Mission Planner Ground Station software was employed. After the hardware assembly and wiring, the standard setup was performed, i.e., (custom) firmware installation, accelerometer and compass calibration, radio configuration, and calibration, and ESC calibration.

3.3. Dataset acquisition

The dataset consists of 18 flights, that were performed in open air in favorable climatic conditions (i.e., mild wind). Each flight consisted of a take-off, flying over four traffic cones that delimit a square, and then landing. The UAV was piloted manually by a trained operator and without any strict requirement on the path to be followed or the flight time. The pilot could decide to hover in place or to perform additional maneuvers, to simulate obstacle avoidance or focusing on a fixed spot. Moreover, the battery State Of Charge (SOC) was different at the beginning of each flight. This makes the dataset more realistic and challenging. In fact, the battery SOC is not guaranteed to be 100% in real flight conditions, and manual flight is often preferred in many applications. Autonomous flights are more reproducible, so it is easier to perform FD in such a case. In other words, using a predetermined trajectory and a fully charged battery makes data more predictable and less realistic, leading to a FD strategy that cannot be generalized to real flight conditions.

To inject faults in the actuators, mechanical damage to the propeller was considered. Blade chipping is a common damage in UAVs as a consequence of the impact of the blade with obstacles. Hence, two kinds of damaged propellers were realized, by chipping a single blade of the propeller (see Fig. 2). Given a blade length of 11 cm, we denote with:

- 0% fault: a propeller with two regular blades whose length is 11 cm;
- 5% fault: a propeller with a regular blade and a chipped blade whose remaining length is 10.45 cm;
- 10% fault: a propeller with a regular blade and a chipped blade whose remaining length is 9.9 cm.

[1] https://firmware.ardupilot.org/Copter/stable/CubeBlack/, last access: 15-12-2022.
[2] https://www.eclipse.org/downloads/, last access: 15-12-2022.
[3] https://www.cygwin.com/install.html, last access: 15-12-2022.
[4] https://firmware.ardupilot.org/Tools/STM32-tools, last access: 15-12-2022.

Fig. 2. Blade chipping. From top to bottom: healthy blade, 5% fault (artificially chipped blade), 10% fault (artificially chipped blade), chipped blade due to a crash.

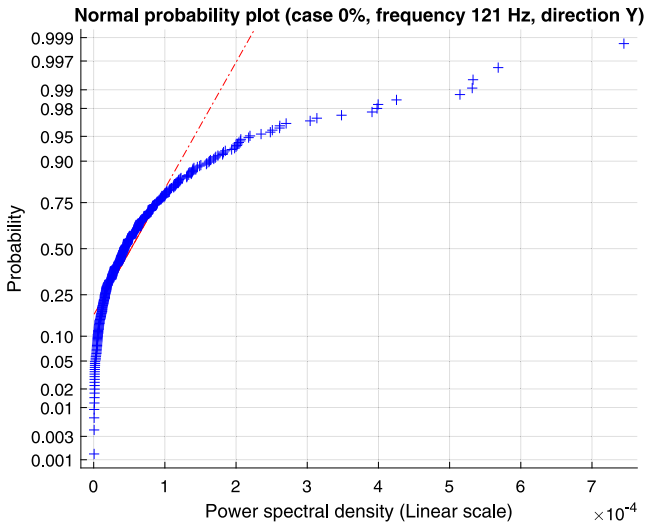


Fig. 3. Normplot of power spectral density (case 0%, frequency 121 Hz, direction Y); data distribution in blue plus signs, normal distribution in red dashed line.

As the propeller profiles differ according to the ClockWise (CW) or CounterClockWise (CCW) rotation direction, a total of four propellers were damaged to create the complete dataset, i.e.:

- a CW propeller with 5% fault,
- a CCW propeller with 5% fault,
- a CW propeller with 10% fault,
- a CCW propeller with 10% fault.

Three different flight scenarios were considered, for a total of 18 flights. Six flights were performed in nominal conditions, i.e., in absence of propeller faults. Then, the 5% fault was injected by installing the proper chipped propeller, one at a once, on each motor, obtaining six flights with a single fault. Similarly, six flights were recorded injecting a single 10% fault. Please note that both the 5% and 10% faults are relatively small (for the hexarotor under investigation) and so they have a negligible impact on the flight dynamics: for this reason, the pilot was still able to fly the drone, and it was still possible to carry out the tests.

4. Preliminary data analysis

In this Section, a hypothesis testing is carried out to evaluate the statistical differences in the frequency domain of the acceleration signals between the healthy case and the two faulty cases.

4.1. Data preprocessing

As anticipated in Section 3.2, the PixHawk flight controller is not a hard real time controller: despite the desired logging frequency is set to 350 Hz, the data is logged at an actual average rate of 344–345 Hz. Moreover, data is not evenly spaced in time. Two approaches have been tested to synchronize the acceleration signals at 350 Hz for frequency domain analysis. The first one is based on the implementation of a Zero-Order Hold (ZOH) model that holds its input for the specified sample period. The second one is based on the resampling of the collected nonuniform data, which consists of upsampling, linear interpolation at the new query points, applying a FIR antialiasing filter to the upsampled signal, and finally discarding samples to downsample the filtered signal.

The experimental results obtained by the two approaches are comparable, so, in the remainder, the proposed results are related to the ZOH-based method, since it requires a lower computational burden.

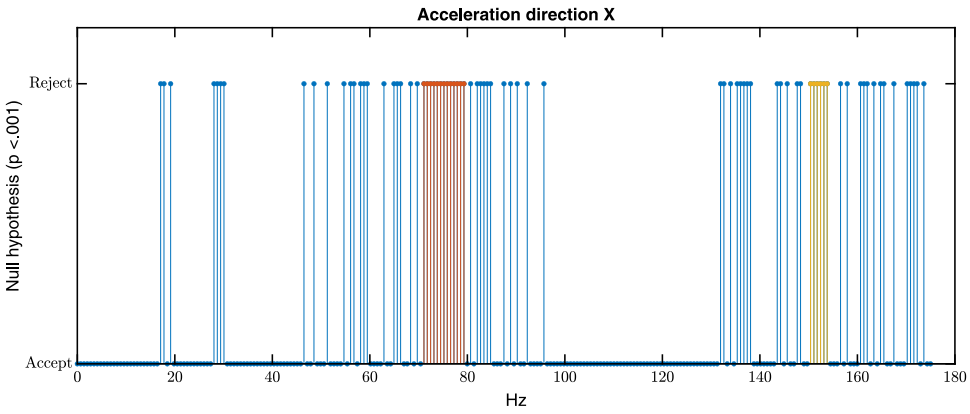
The take-off and landing phases are not analyzed and the corresponding signals have been neglected, as well as the data related to the UAV when on the ground. Such phases can be recognized, for example, by looking at the commanded Pulse Width Modulation (PWM) signal to the motors (e.g., 1100 when landed and idle).

4.2. Statistical data analysis

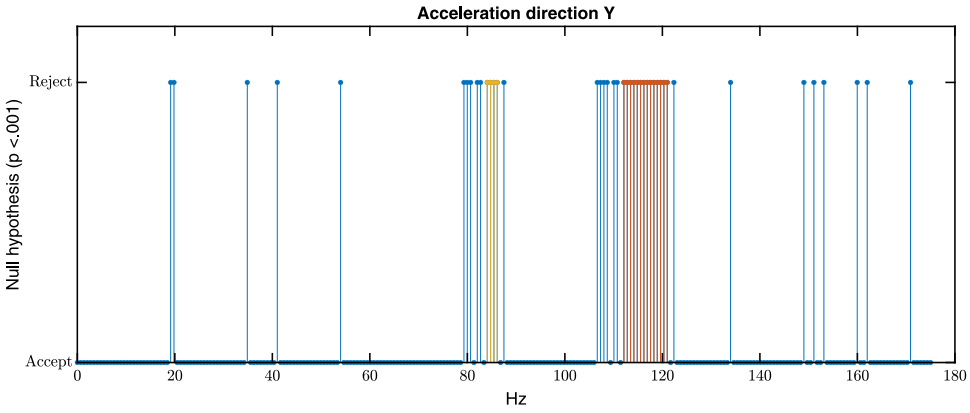
The dataset is partitioned into windows of 1 s length without overlapping. In each time window (i.e., 350 samples), we perform a FFT with 512 points, using zero padding and a Hamming window. Please note that this preliminary analysis is performed only with the aim of investigating the frequency components in different time windows, and at different flight (fault) conditions, to ensure that a frequency-based FD method is meaningful; FFT will not, however, be used for feature extraction and FD design.

Consider the flight in absence of faults only: we note that most of the frequencies show a nongaussian distribution, i.e., more than 95% of the frequencies deviate from normality according to the Anderson–Darling test ($p < 0.01$). Similar results hold in presence of a 5% or a 10% fault. As an example, a normal probability plot is reported in Fig. 3. It shows the distribution of the Power Spectral Density (PSD) computed at a single frequency (121 Hz) for the faultless case (0% fault), in the Y axis acceleration, compared to the normal distribution. The poor fitting using a gaussian approximation is evident.

As gaussianity cannot be assumed, the non-parametric Kruskal–Wallis (KW) test is considered to verify if the data in the frequency domain, taken from different fault conditions, are drawn from the same distribution. In other words, we test if some frequencies show a different pattern in presence of a fault, in order to define features for FD. A multiple comparison test on PSDs is carried out to provide information about which pairs of medians are significantly different, and which are not. The null hypothesis has been tested with a significance level set to 0.001. Fig. 4 shows the results of the KW test. For each accelerometer, the plots denote whether each frequency is relevant for FD (i.e., it shows a significantly different distribution in case of faults, so the null hypothesis is rejected) or not (null hypothesis is accepted). The Z axis is omitted because the null hypothesis is accepted for every frequency. We note that, in both cases, there are two major clusters of relevant frequencies: they are highlighted in red and in orange, respectively. The signals in such coherent bands are easy to be extracted online by means of a Band-Pass Filter (BPF), thus avoiding to employ FFT online.



(a) X direction.



(b) Y direction.

Fig. 4. Kruskal–Wallis KW test results.

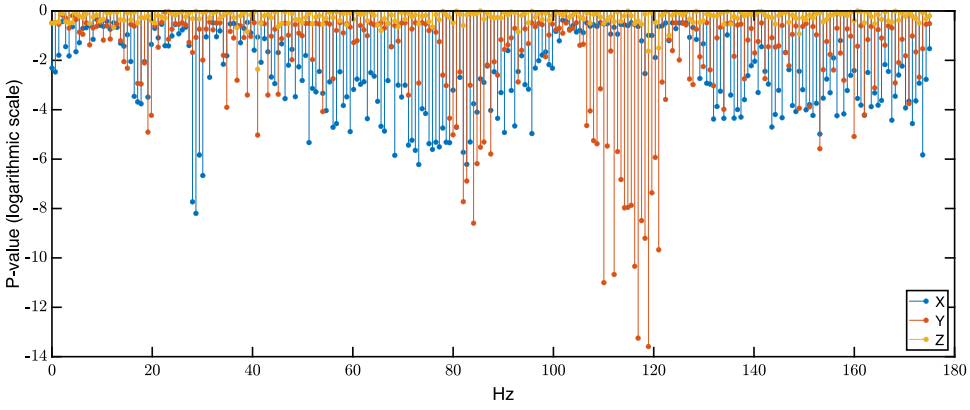


Fig. 5. P-values in logarithmic scale of direction X, Y, Z.

Fig. 5 reports the average p-values obtained by the multiple comparison test. We note that the band centered in 115 Hz for the Y acceleration is associated with the most significant p-values, followed by the lower band centered in 90 Hz, especially in Y as well. The band around 100 Hz shows poor significance as it is related to the main rotational speed in hovering. Moreover, the vibration in the Z axis shows in general a poor significance: in fact, we expect the vehicle to oscillate mainly in the horizontal plane due to unbalancing.

Also please note that the median rotation speeds of the propellers lie in $80-100$ Hz, in accordance with the speed measurement provided by the ESC through back-EMF (see Fig. 6). The different mean speed of the motors is not determined by the faults on the propellers, as the same pattern can be noticed in the fault-free flights only. Moreover, the dataset is balanced, and all of the motors are faulty for the same number of flights, as detailed in Section 3.3. The difference in motor speeds can be ascribed to both the asymmetrical distribution of payloads and, possibly, to the particular flight trajectory (e.g., yawing CW more than CCW).

More specifically, the PSD in the meaningful bands increases in presence of a fault, as shown in Fig. 7. Such increased vibration in presence of a fault is physically justified by the unbalancing in the rotating propeller. Although the propeller mass is very small if compared to the entire UAV, an unbalanced rotating propeller moves the center of mass according to its current orientation. Moreover, the shape of the broken tip can also cause additional turbulence and thus vibrations. Considering the Z axis instead (Fig. 7(c)), the PSD

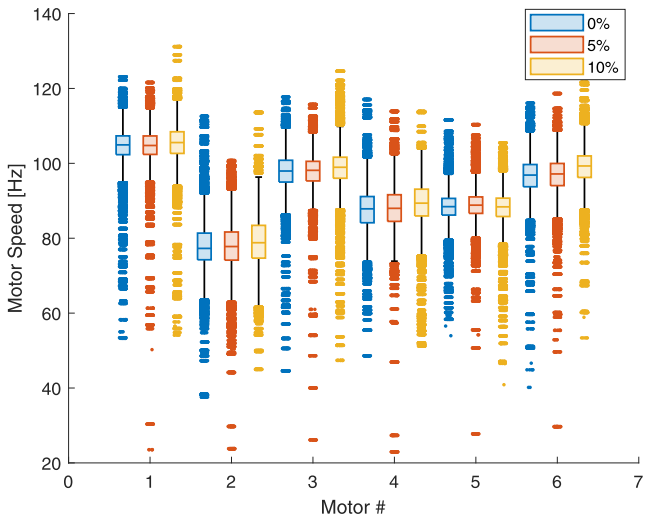


Fig. 6. Probability distribution for the motor speed measurements.

apparently increases around 100 Hz. This can be justified by the loss of lift caused by the fault, which can slightly alter the average rotational speed: in fact, we expect the flight controller to compensate for the thrust loss by slightly increasing the rotational speed of the faulty motor. However, the statistical analysis has shown poor significance. In fact, we also expect maneuvering to affect the vertical lift force and thus the vibration, so the feature could be unreliable for the purposes of FD.

Also note that disregarding the presence of faults, a peak in power spectral density can be noticed in the range of 80–100 Hz (clearly visible in X, Fig. 7(a)), which is related to the average motor speed. Additional relevant frequencies for FD can be found in Fig. 4, but they lack cohesion, so their extraction without involving not significant features require selective (and thus more complex) filters.

Fig. 8 reports the boxplot of the power spectral densities in two meaningful bands. Please note that the logarithmic scale, which is adopted for the sake of readability, graphically reduces the distance between the outliers and the median: the same data, plotted on a linear scale, is characterized by the presence of several huge outliers. According to the statistical analysis, we propose a filtering-based approach consisting of the design of a bank of filters to extract frequency-based features thus avoiding the implementation of the FFT on an embedded board.

5. AI-based fault detection design

In this Section, we design two alternative AI-based FD algorithms. In the first case, we employ the default features suggested by MATLAB Diagnostic Feature Designer (DFD), i.e., 261 features including time and frequency domain. The DFD is a tool to help by graphical user interface with the automatic feature extraction of time-series for diagnostic and predictive purposes. In the second case, we employ a set of frequency-domain features, which are extracted by means of FIR filters, without the need of computing any FFT online.

5.1. Classifier training

For a fair comparison, in both cases we train a classifier, in a supervised manner and using the same procedure, to discern between three different groups (i.e., 0%, 5%, and 10% fault cases). To limit the computational effort, sparse classifiers are designed. As in sparse classifiers many features are not employed (or they have a null weight), only a subset of the features must be actually calculated online and fed to the classifier. The size of the such subset of features, which determines the complexity, depends on the classifiers' hyperparameters, so it can be tailored to the available computational power. The dataset is divided into two partitions, 70% for training and 30% for testing, with stratification (i.e., preserving the same proportions between groups in the test set) to avoid sampling bias. The partitioning is performed 20 times, where the training and testing datasets are randomly partitioned and, for each run, the same kind of classifier is trained. A 5-folds cross-validation is employed to prevent overfitting, and the main hyperparameters are explored through a grid search.

Avoiding false positives is a priority in the aeronautical field, as false positives usually trigger safety procedures that must be avoided when unnecessary (e.g., reconfiguration, emergency landing). Hence, the following matrix is employed to compute the misclassification cost:

$$C = \begin{bmatrix} 0 & 10 & 10 \\ 1 & 0 & 1 \\ 5 & 1 & 0 \end{bmatrix}, \tag{1}$$

where the row represents the true class (from the top, 0%, 5%, 10% fault, respectively) and the column represents the predicted class (from the left, 0%, 5%, 10% fault, respectively). The proposed misclassification cost matrix weighs false positives (true class 0% fault, wrong predicted class) with a factor of 10, while a factor of 5 is employed to weigh false negatives in case of the most severe fault (true class 10% fault, predicted class 0% fault). The unitary weight is adopted for any other misclassification.

5.2. DFD-based design

The DFD-based detection approach basically uses the feature extraction step from the DFD. We calculate a total of 261 features by using the default ones proposed by the DFD in MATLAB. As shown in Table 1, we report the 28 time-domain features and the 3 frequency-domain features. The time-domain features are calculated for the raw signals, the first-order difference of raw signals, and the residual signals (evaluated as the raw signals minus the ensemble mean), for a total of 84 time-domain features. Hence, 87 features are calculated for each acceleration signal (i.e., X, Y, and Z).

5.3. Filtering-based design

The filtering-based approach consists in designing a bank of equiripple FIR filters with a minimum order to extract frequency-based features. In order to maximize the performances, many frequency-based features are proposed in the following.

Taking into account the statistical analysis in Section 4, where the lower frequencies do not show sufficient statistical significance, we design a set of band-pass and high-pass FIR filters. The filter properties are detailed in Table 2.

In particular, 32 High-Pass Filters (HPFs) are designed, where the stopband frequency is set to

$$5, 10, \ldots, 155, 160 \text{ Hz}, \tag{2}$$

respectively. As for the BPFs, 22 filters are designed. We design 15 filters characterized by a 10 Hz fixed bandwidth, where the first stopband is set to

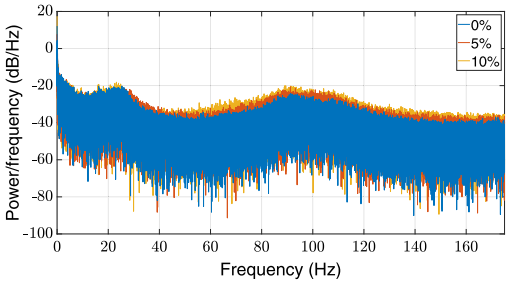
$$5, 15, \ldots, 135, 145 \text{ Hz}, \tag{3}$$

respectively, so that the second stopband is set to

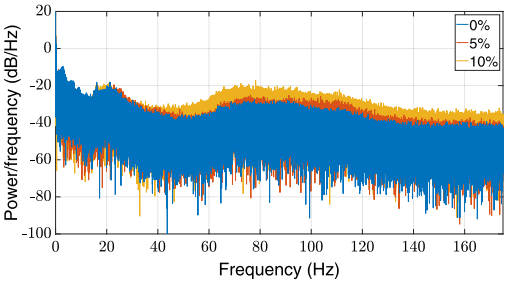
$$15, 25, \ldots, 145, 155 \text{ Hz}, \tag{4}$$

respectively. Moreover, 7 additional BPFs characterized by a 20 Hz fixed bandwidth are added, where the first stopband is set to

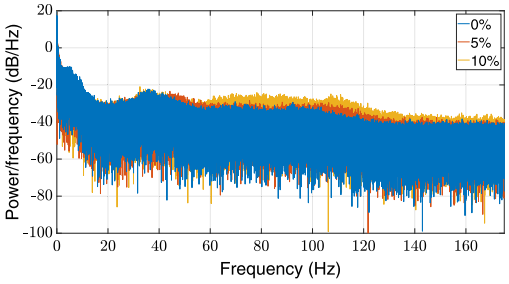
$$5, 25, \ldots, 105, 125 \text{ Hz}, \tag{5}$$



(a) X direction.

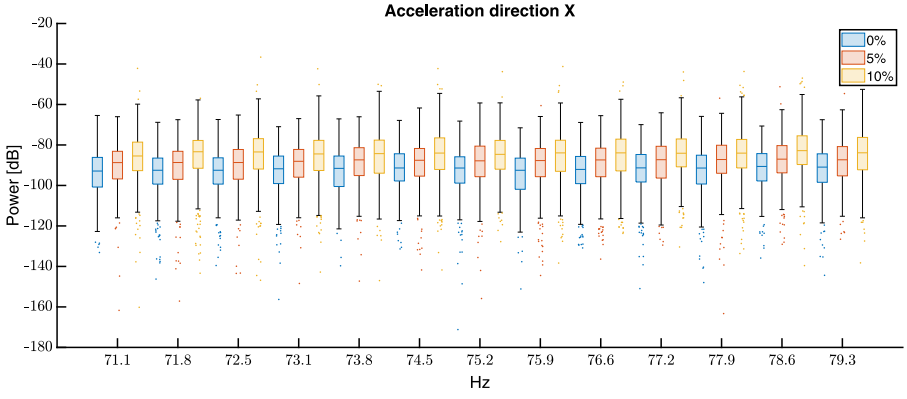


(b) Y direction.

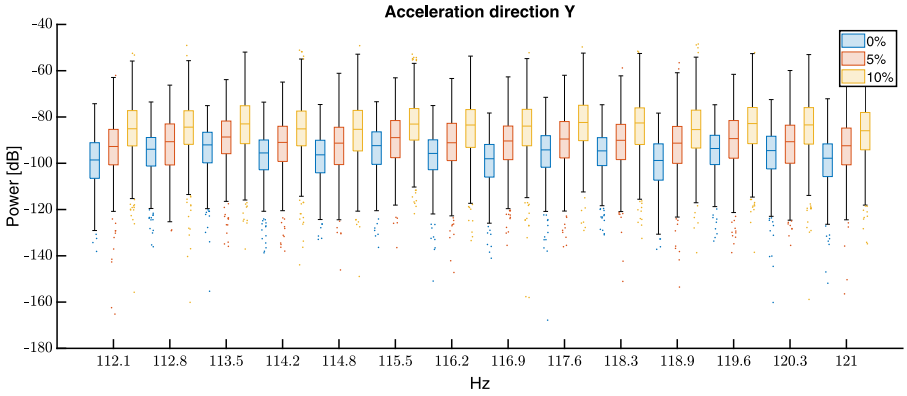


(c) Z direction.

Fig. 7. Acceleration periodogram.



(a) X direction.



(b) Y direction.

Fig. 8. Boxplot of power spectral densities.

Table 1
List of features extracted by DFD app.

#	Domain	Set of features	Subset of features	Features
1	Time	Signal	Statistical	Mean value
2	Time	Signal	Statistical	Standard Deviation
3	Time	Signal	Statistical	Skewness
4	Time	Signal	Statistical	Kurtosis
5	Time	Signal	Statistical	Root Mean Square
6	Time	Signal	Statistical	Shape Factor
7	Time	Signal	Impulsive	Clearance Factor
8	Time	Signal	Impulsive	Crest Factor
9	Time	Signal	Impulsive	Impulse Factor
10	Time	Signal	Impulsive	Peak Value
11	Time	Signal	Harmonic	Signal-to-Noise and Distortion Ratio
12	Time	Signal	Harmonic	Signal-to-Noise Ratio
13	Time	Time Series	Distribution	Minimum
14	Time	Time Series	Distribution	Median
15	Time	Time Series	Distribution	Maximum
16	Time	Time Series	Distribution	Quartile Q1
17	Time	Time Series	Distribution	Quartile Q3
18	Time	Time Series	Distribution	Interquartile range
19	Time	Model-based	Autoregressive Model	Model Coefficients (1°)
20	Time	Model-based	Autoregressive Model	Natural Frequencies (1°)
21	Time	Model-based	Autoregressive Model	Damping Factors (1°)
22	Time	Model-based	Model Fit	Mean Squared Error
23	Time	Model-based	Model Fit	Mean Absolute Error
24	Time	Model-based	Model Fit	Akaike's Information Criterion
25	Time	Model-based	Model Residual	Residual Mean
26	Time	Model-based	Model Residual	Residual Variance
27	Time	Model-based	Model Residual	Residual RMS
28	Time	Model-based	Model Residual	Residual Kurtosis
1	Frequency	Spectral	Spectral Peaks	Peak amplitude
2	Frequency	Spectral	Spectral Peaks	Peak frequency
3	Frequency	Spectral	Band Power	Band Power

Table 2
FIR filters specifications.

Specification	Band pass	High pass
First stopband attenuation	30 dB	30 dB
Second stopband attenuation	30 dB	n.a.
Filter bandwidth	10 Hz/20 Hz	n.a.
Passband ripple	1 dB	1 dB
Density factor	20	20

respectively, so that the second stopband is set to

$$25, 45, \ldots, 125, 145 \text{ Hz}. \tag{6}$$

Each acceleration signal (X, Y, and Z) is filtered using the 54 filters that have been detailed, thus obtaining 162 filtered signals. For each filtered signal, we calculate the following features:

- mean value,
- peak value,
- variance,
- mean of absolute values,
- mean of squared values.

Hence, 810 features are employed to train the FD classifiers, and then sparsity is enforced to decrease the computational effort.

5.4. Computational complexity

Running each filter with a single input value requires a time complexity $\mathcal{O}(N)$, where N is the order of the filter. This operation is repeated whenever a new acceleration sample is available. Let W be the number of samples for each signal in the time window (i.e., 350 in 1 s), n_s the number of signals to be filtered (i.e., 3), and n_f the number of filters (i.e., up to 54). Then, the overall time complexity to calculate the filtered signals in a single (1 s) time window is $\mathcal{O}(n_f \cdot N \cdot n_s \cdot W)$. In other words, once the filters are fixed, the time complexity is linear with respect to the number of acceleration samples. Experimentally, the computation complexity of buffering the signal for 1 s and then calculating the features is too high to be performed online (i.e., more than 2.5 ms, thus infeasible in a single scheduler iteration). Therefore, filtering and calculating the features is necessarily performed online in an iterative way, i.e., by updating the result as soon as a new acceleration measurement is available. This allows to split the overall computation time from a single iteration ($\mathcal{O}(n_f \cdot N \cdot n_s \cdot W)$) to many ($W$) smaller computations ($\mathcal{O}(n_f \cdot N \cdot n_s)$) performed in W different iterations. Also, using an iterative algorithm to calculate the features, the output of the filters does not need to be stored for the whole time window, thus reducing the space complexity. The five features listed in Section 5.3 can be calculated iteratively, and then the values after 1 s are employed for FD. The online update of mean and variance is carried out by using the formulas for the computation of higher-order central moments defined by Pebay et al. (2014), Pébay et al. (2016), Meng (2015). Each of the five feature calculations is linear in the number of input samples, and the time complexity of each one is $\mathcal{O}(n_f \cdot n_s)$ for each of the W time intervals where a new acceleration signal is measured.

In case of a Linear Support Vector Machine (LSVM) classifier, the additional classification time complexity is $\mathcal{O}(n_z)$, where n_z is the number of features with a non-zero weight, as the decision boils down to a weighted sum and a comparison with a fixed threshold. In the case of a binary classification decision tree, instead, the additional classification

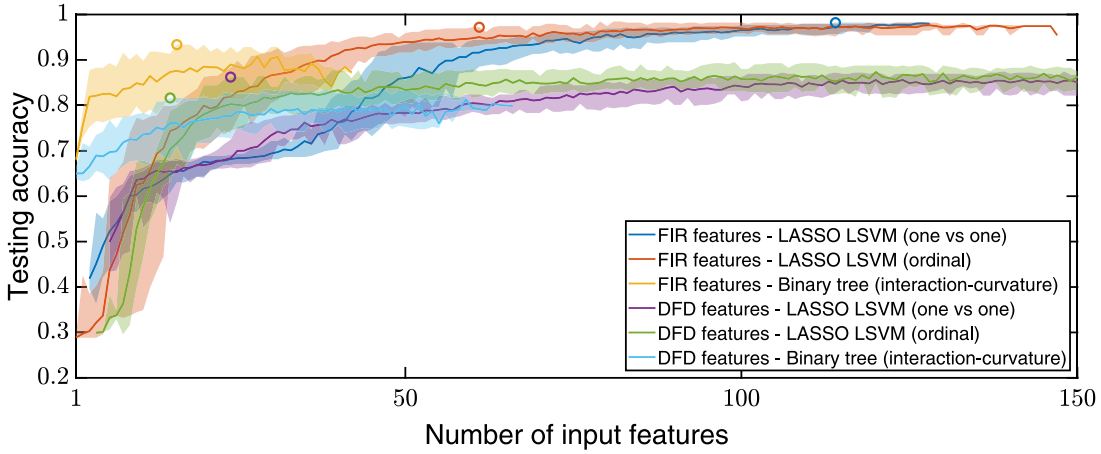


Fig. 9. Accuracy versus the number of actually employed features.

time complexity is $\mathcal{O}(h)$, where h is the tree height. Please note that the height of a balanced tree is logarithmic in the number of nodes, and thus it is logarithmic in n_z, i.e. the number of features that are used to make the splits. Thus, given a fixed number of features n_z, we expect the classification tree to be faster than the classification through a LSVM.

6. Experimental results

In this Section, we compare the results using the algorithms designed in Section 5, i.e., DFD-based features versus FIR filtering-based features.

6.1. Filtering-based versus DFD-based detection

For a fair comparison, we train several sparse classifiers, i.e., LASSO LSVMs with a one versus one classification method, LASSO LSVMs with an ordinal classification method, and binary classification trees using an interaction-curvature training method. For each of them, we explore the main hyperparameter that regulates sparsity. In the case of LASSO LSVMs, the regularization factor λ, that weights the L1 penalty ($\lambda \sum_j |\beta_j|$, where β_j is the weight associated to the jth feature), determines the sparsity. Hence, λ is explored using 300 logarithmically spaced tentative values, from 10^{-3} to $10^{-0.5}$. In the case of binary trees, the maximum number of splits ("MaxNumSplits") hyperparameter is explored, as it represents an upper bound for the number of features that are actually employed for classification.

Fig. 9 reports the accuracy versus the number of actually employed input features (i.e., an indicator for time complexity). As anticipated in Section 5.1, 20 runs are performed with a different partitioning to show if the performances are consistent: the solid line represents the average accuracy, while the shaded area of the same color defines the interval between the best and the worst accuracy using the same number of features over the 20 runs.

First of all, we note that, given a classifier, the FIR features outperform the DFD features for every number of input features and for every classifier. Not only the average value using the FIR features are consistently above the average using the DFD features and the same classifier, but also the shaded areas are often separated (e.g., in the case of a binary tree classifier, the yellow FIR based area is above the green DFD based area). The results also show that, for both FIR and DFD features, the binary trees outperform the LSVMs for a very low number of features.

Given the evident superiority of the FIR based features, from now on, we focus on the analysis of FIR based FD algorithms. We note that, with just three features and a binary classifier using the FIR features (namely, the peak of the y-axis accelerations after a 75 Hz and a 125 Hz HPF, respectively, and the mean of the squared values of the x-axis accelerations after a 60 Hz HPF), the accuracy is up to 87.76%. For FIR features, the breakeven with the ordinal LSVM occurs when at least 34 features are kept: in fact, the performances of the tree classifiers saturate after approximately 20 features, while the LSVMs show to be more effective when a larger amount of features is available. Comparing the ordinal and the one versus one LSVMs, we note that the one versus one can achieve marginally better results at the price of increasing the number of features. In fact, the ordinal strategy requires 2 LSVM classifiers to separate three classes, while the one versus one employs 3 classifiers, so it may require several additional features for training. No classifier benefits from using more than 100 features, and the accuracy saturates in the 95–98% range in the case of LSVMs using FIR features.

6.2. Filtering-based FD performances

In the following, we detail the performances in the testing stage of three specific classifiers that are depicted in Fig. 9 with a circle. We recall that the test set is obtained using a 70%–30% partitioning with stratification.

The confusion matrix in Fig. 10(a) refers to a binary tree classifier that uses 16 FIR based features and it shows a 93.37% accuracy in testing (yellow circle in Fig. 9). The false positive rate, i.e., the detection of a fault when the true class is no fault, is 11.5%. The classifier has been obtained by setting the hyperparameter "MaxNumSplits" to 18. To calculate the 16 input features, 14 filters are actually involved (i.e., some filtered quantities are used more than once, e.g., considering the output of the HPF with a 60 Hz stopband on the X acceleration, both the mean and the mean of the squared values are actually employed by the classifier). To analyze the consistency of the result, we note that, using 16 features, the average accuracy (i.e., the solid yellow line in Fig. 9) of the trained binary tree classifiers is 87.59% and the standard deviation is 3.62%.

The confusion matrix in Fig. 10(b) refers to an ordinal LSVM that employs 61 features and it shows a 97.19% accuracy in testing (red circle in Fig. 9). The false positive rate is reduced to 0%. The classifier has been obtained by setting the hyperparameter $\lambda = 9.8852 \cdot 10^{-3}$. To calculate the 61 input features, 43 filters are actually involved. The average accuracy of the ordinal LSVMs that employ 61 features is 95.03% and the standard deviation is 1.04%.

The confusion matrix in Fig. 10(c) refers to a one versus one LSVM that employs 114 features and it shows a 98.21% accuracy in testing (blue circle in Fig. 9). The false positive rate is 1.8%. The classifier has been obtained by setting the hyperparameter $\lambda = 1.2844 \cdot 10^{-3}$. To calculate the 114 input features, 64 filters are actually involved. The

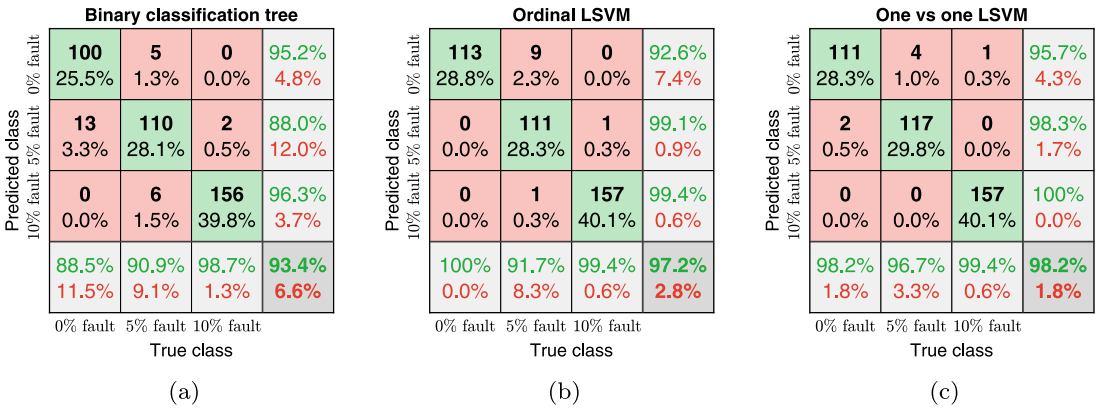


Fig. 10. Confusion matrices in testing (FIR-based features).

Table 3
Classification accuracy and number of input features for the DFD-based and filter-based FD strategies.

Features	# Features	# Filters	Classifier	Accuracy Best	Avg	Std
DFD	15	n.a.	tree	81.63%	76.10%	2.89%
DFD	24	n.a.	ordinal LSVM	86.22%	80.27%	2.57%
DFD	106	n.a.	1 vs 1 LSVM	87.24%	84.55%	1.69%
filters	16	14	tree	93.37%	87.59%	3.62%
filters	61	43	ordinal LSVM	97.19%	95.03%	1.04%
filters	114	64	1 vs 1 LSVM	98.21%	97.21%	0.58%

Table 4
HIL execution time and number of input features the filter-based FD strategies.

Features	# Filters	# Features	Task	HIL execution time Average
filters	70	140	Feature computation	0.0158 ms
filters	70	140	LSVM classification	0.0154 ms
filters	70	140	Total over 1 s window	5.55 ms

average accuracy of the one versus one LSVMs classifiers that employ 114 features is 97.21% and the standard deviation is 0.58%.

The proposed results are summarized in Table 3 where, given a number of features that show a good trade-off between complexity and accuracy (i.e., the same number of features considered for Fig. 10), the best, the mean, and the standard deviation of the accuracy is reported. For the sake of comparison, we also report in Table 3 the same quantities in the case of DFD based features. The classifiers with the best accuracy are depicted in Fig. 9 with a circle.

6.3. Hardware-in-the-loop complexity testing

The FD algorithms are tested in HIL using Matlab's UAV toolbox, which allows to design the algorithm in Simulink and then to compile and deploy it to a target flight controller. The target hardware for the deployment is a Pixhawk 2.1 Cube black, equipped with a 32-bit ARM Cortex M4 core, 168 MHz, 256 KB RAM and 2 MB Flash.

Please note that the Pixhawk 2.1 Cube black is a legacy hardware with limited computational power and memory, and newer alternatives are available off-the-shelf. Indeed, the following expedients are needed to run the algorithm online on the target device. First of all, storing the acceleration data for 1 s, and then performing batch filtering and calculation of the features, has clearly shown to be infeasible for both time and space complexity reasons. This makes impossible to deploy the DFD based features, because most of them are too complex, and also many of them cannot be efficiently implemented in an iterative way. On the other hand, the filtering-based features can be calculated iteratively, and then sampled after 1 s to be fed to the FD classifier. To optimize the space complexity, all the floating point variables are stored in single precision (IEEE 754 standard), while the integers are stored using a 16-bit representation. Then, the amount of memory to store data is approximately 30 KB, where the largest part is due to the FIR filters' state. To decrease the computational time, all the buffers (i.e., the states of the FIR filters) are implemented as circular buffers. Fig. 11 reports the actual implementation of the calculation of the iterative features and the binary tree classifier in Fig. 10(a). The diagrams for the implementation of Figs. 10(b) and 10(c) can be inferred and they are omitted for brevity.

To test the time complexity, the algorithm to calculate the features and the classification is run during a flight simulated in HIL. The Simulink algorithm for the joint flight control and FD are deployed to the target hardware, the Qgroundcontrol.[5] is employed to define the flight mission, while the UAV Dynamics model runs in Matlab to simulate the dynamics of the UAV For this purpose, we test the computational time implementing 70 filters and 140 features, which is a more demanding task than the most complex classifier in Table 4. Then, the time to perform the FD computations is recorded. We note that the computation time is small compared to the resolution (1 ms) using the Posix time.h library on the flight controller. Hence, the FD computations are repeated more than once in every iteration to increase the computation time, and the time is inferred by linear regression.

Fig. 12 reports the experimental HIL time to compute the FIR filtering and to extract the features. In the abscissa, the number of repeated (identical) computations of filters and features is detailed. We note that the task can be repeated 28 times in the same scheduler iteration without reaching the 1 ms resolution of the time.h library, while the median time is less than 1 ms up to 48 repetitions. As the slope of the fitted line is $1.58 \cdot 10^{-5}$, we infer that the average time to filter and calculate the features is 15.8 μs. This task must be then repeated 350 times per second.

Fig. 13 reports the experimental HIL time to run the LSVM classifier with 70 features. Again, we note that the task can be repeated 40 times in the same scheduler iteration without reaching the 1 ms resolution of the time.h library. The slope of the fitted line is $1.54 \cdot 10^{-5}$, so we infer that the average time to run the classifier is 15.4 μs. The classification task is run only once per second. In Table 4, we report the mean execution time in HIL.

According to Table 4, the mean execution time to perform the feature update in flight is 0.0158 ms, while 0.0154 ms are needed to calculate the classification result via LSVM. In the case of a binary tree, instead of a LSVM, the time for classification is negligible, consistently with the discussion in Section 5.4. Please note that the features are

[5] http://qgroundcontrol.com/, last access: 27-03-2023.

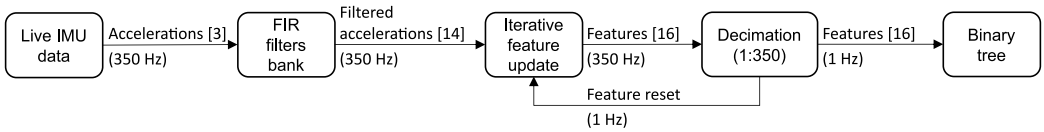


Fig. 11. Iterative features calculation and binary tree classification.

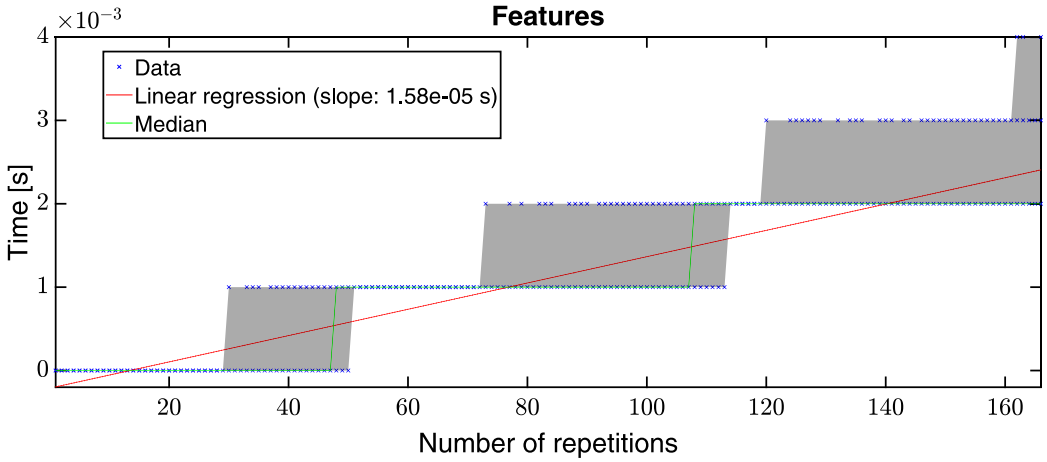


Fig. 12. FIR filtering and feature extraction (70 filters, 140 features): HIL execution time on a Pixhawk Cube black.

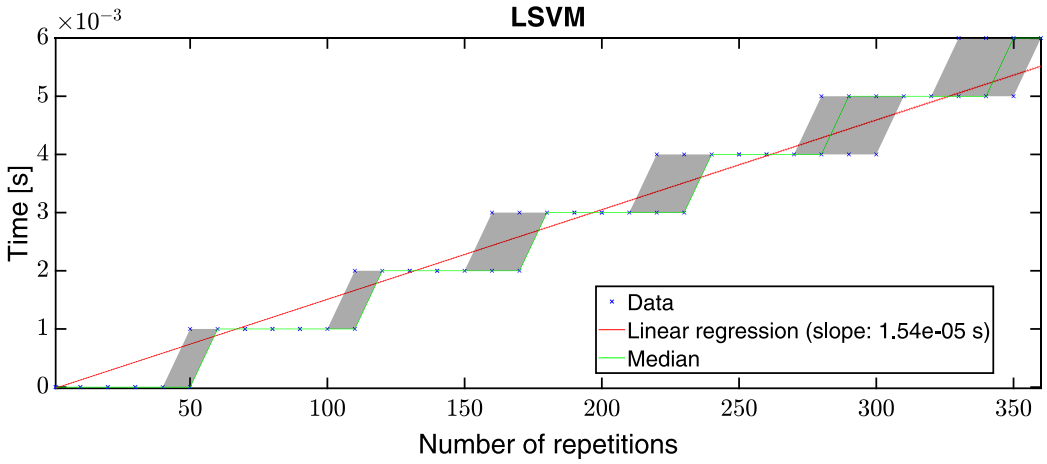


Fig. 13. LSVM classification (140 features): HIL execution time on a Pixhawk Cube black.

updated iteratively at 350 Hz, i.e., when a new acceleration signal is available, while the classification task is performed only once per 1 s time window, i.e., at 1 Hz frequency. So, the overall computation time for each 1 s time window is 5.55 ms, that is 0.55% of the CPU time on average. The maximum computational load is achieved once per second, in the final frame, when both the feature update and the LSVM computation occur, for a total of 0.0312 ms on average. Comparing it to the scheduled time to execute all the tasks in each cycle at 400 Hz (i.e., 2.5 ms), we can state that all of the three proposed FD algorithms can safely run on the target hardware, as it takes less than 2% of the cycle time in the most computationally intensive iteration.

6.4. Testing with a chipped blade due to a crash

In order to validate the behavior of the proposed FD strategies in real scenarios, we test the three FIR filtering-based classifiers from Table 3 using a different blade with a fault due to a crash. We remark that this additional flight data is not employed in the training stage. The profile of the blade has been shown in Fig. 2. Please note that the shape of the blade is different from both the healthy and the 5% and 10% blades that have been artificially damaged instead. The severity of the fault is deemed as intermediate, i.e., more than 5% and less than 10%. In Fig. 14, we report the output of the three classifiers during a flight whose duration is 125 s, hence 125 classification outputs are available. For each time window (1 s) of the test set, the abscissa represents the window number and the ordinate is the classification result. The testing windows are sorted in ascending order by the acquisition time, hence subsequent windows in the same flight are adjacent. As the blade fault is intermediate, we consider 5% and 10% fault as acceptable answers, while 0% clearly represents a false negative. Under this assumption, the accuracy of the three classifiers in testing is reported in Fig. 14 as well. All of the proposed classifiers warn that a fault is occurring, but we note that the best performances are achieved by the binary tree, which is the most simple and with the lowest accuracy in training, with a striking 100% accuracy over 125 time windows. As the binary tree selects only a few (i.e., 16) relevant features, it avoids overfitting better than the more complex LSVMs, thus showing good performances in real tests. The accuracy of the LSVM classifiers is actually lower than

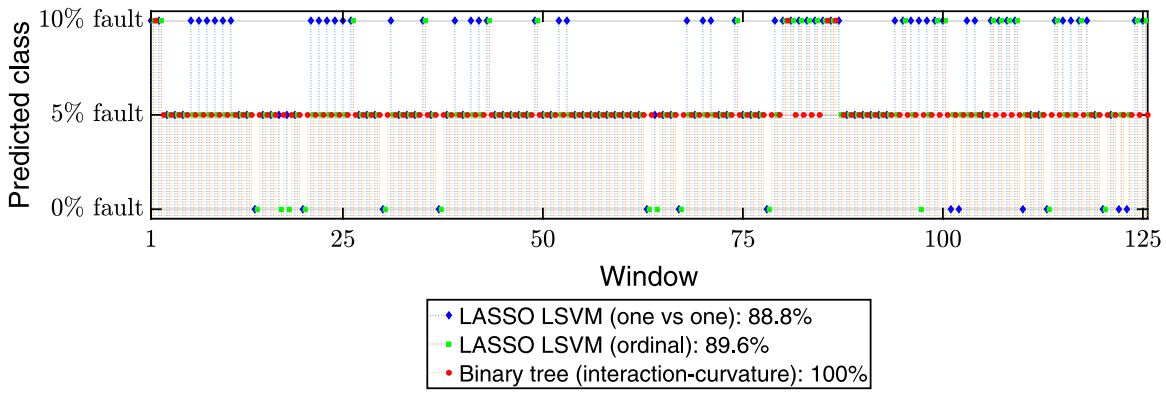


Fig. 14. Classification results: testing with a chipped blade due to a crash.

in Table 3, as they can miss the presence of the fault, which means that the detection delay can increase.

7. Conclusion

Propeller faults on multirotor drones may have a severe impact on vehicle dynamics. In this study, we have investigated subcritical propeller faults, that have minor effects on the dynamics of the hexarotor under investigation. Nonetheless, such faults can be detected with high accuracy, starting from the on-board IMU data and simple classifiers, while keeping the computational effort small enough to be run in real-time on a conventional flight controller with severe computational constraints.

We have found that calculating the features in a batch is infeasible for both time and space complexity reasons. This makes impossible to deploy the DFD based features, as they cannot be efficiently implemented in an iterative way.

On the contrary, FIR filtering-based features have proven to be feasible online even on a legacy flight controller. We obtain up to 93.37%, 97.19%, and 98.21% accuracy using a classification tree (16 input features), an ordinal LSVM (61 input features), and a one versus one LSVM (114 input features), respectively.

The HIL tests show that the FD algorithm is feasible, with less than 2% of the cycle time to be dedicated to FD in the worst case, and a 0.55% of average CPU time. Moreover, the tests with a different faulty blade, that has been damaged due to a crash, show that the results are consistent in real scenarios, with a 100%, 89.6%, and 88.8% accuracy using the classification tree, the ordinal LSVM, and the one versus one LSVM, respectively. The main challenge, in view of actual implementation in the field of UAVs, is to keep a small false positive rate.

The limitations of the proposed results can be identified in neglecting the external disturbances (in particular, the wind) and the need for adapting the features (especially for vehicles with a different size) and thus retraining the classifier. Although the proposed algorithm is validated in realistic flight conditions, i.e., using manual flight mode and a different, unknown battery SOC, the fact that the algorithm is not tested in presence of wind represents a current limitation. However, as the features are extracted at high frequencies, we expect that the influence of external wind on these frequencies is marginal, as well as the maneuvers performed by the pilot because they both affect the lower frequencies mainly. Finally, as the proposed approach relies on vibration at specified frequencies, we expect the classifier should be retrained on new vehicles, especially when the average rotational speed of the propellers is substantially different from the UAV under investigation, e.g., due to a different UAV size, blade span, and/or the number of actuators.

In order to increase the performances of the proposed approach, in our future works we are going to investigate the use of Wavelet transform instead of conventional FIR filters. In fact, time–frequency features can be obtained through the Wavelet transform, which could lead to a shorter detection time and increased accuracy. Also, as higher-order central moments can be calculated iteratively as well, we are going to employ skewness and kurtosis to define additional diagnostic features for FD.

List of acronyms

AI Artificial Intelligence.

BPF Band-Pass Filter.

CCW CounterClockWise.

CW ClockWise.

DFD Diagnostic Feature Designer.

ESC Electronic Speed Controller.

FD Fault Detection.

FFT Fast Fourier Transform.

FIR Finite Impulse Response.

HIL Hardware-In-the-Loop.

HPF High-Pass Filter.

IMU Inertial Measurement Unit.

KW Kruskal–Wallis.

LSVM Linear Support Vector Machine.

PSD Power Spectral Density.

PWM Pulse Width Modulation. SOC State Of Charge.

UAV Unmanned Aerial Vehicle.

ZOH Zero-Order Hold.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Ai, Shaojie, Song, Jia, Cai, Guobiao, Zhao, Kai, 2022. Active fault-tolerant control for quadrotor UAV against sensor fault diagnosed by the auto sequential random forest. Aerospace 9 (9), 518.

Baldini, Alessandro, Felicetti, Riccardo, Freddi, Alessandro, Longhi, Sauro, Monteriù, Andrea, 2022. Hexarotor fault tolerant control using a bank of disturbance observers. In: 2022 International Conference on Unmanned Aircraft Systems. ICUAS, IEEE, pp. 608–616.

Benini, Alessandro, Ferracuti, Francesco, Monteriù, Andrea, Radensleben, Stephan, 2019. Fault detection of a VTOL UAV using acceleration measurements. In: 2019 18th European Control Conference. ECC, IEEE, pp. 3990–3995.

Boztas, Gullu, Tuncer, Turker, 2022. A fault classification method using dynamic centered one-dimensional local angular binary pattern for a PMSM and drive system. Neural Comput. Appl. 1–12.

Bronz, Murat, Baskaya, Elgiz, Delahaye, Daniel, Puechmore, Stéphane, 2020. Real-time fault detection on small fixed-wing UAVs using machine learning. In: 2020 AIAA/IEEE 39th Digital Avionics Systems Conference. DASC, pp. 1–10.

Chakrapani, G., Sugumaran, V., 2023. Transfer learning based fault diagnosis of automobile dry clutch system. Eng. Appl. Artif. Intell. 117, 105522.

Chen, Yafeng, Wang, Benkuan, Liu, Wang, Liu, Datong, 2017. On-line and non-invasive anomaly detection system for unmanned aerial vehicle. In: 2017 Prognostics and System Health Management Conference. PHM-Harbin, pp. 1–7.

Fourlas, George K., Karras, George C., 2021. A survey on fault diagnosis and fault-tolerant control methods for unmanned aerial vehicles. Machines 9 (9), 197.

Freddi, Alessandro, Longhi, Sauro, Monteriù, Andrea, 2012. A diagnostic thau observer for a class of unmanned vehicles. J. Intell. Robot. Syst. 67 (1), 61–73.

Gangsar, Purushottam, Tiwari, Rajiv, 2020. Signal based condition monitoring techniques for fault detection and diagnosis of induction motors: A state-of-the-art review. Mech. Syst. Signal Process. 144, 106908.

Ghalamchi, Behnam, Jia, Zheng, Mueller, Mark Wilfried, 2019. Real-time vibration-based propeller fault diagnosis for multicopters. IEEE/ASME Trans. Mechatronics 25 (1), 395–405.

Ghalamchi, Behnam, Mueller, Mark, 2018. Vibration-based propeller fault diagnosis for multicopters. In: 2018 International Conference on Unmanned Aircraft Systems. ICUAS, IEEE, pp. 1041–1047.

Hoang, Duy-Tang, Kang, Hee-Jun, 2019. A survey on deep learning based bearing fault diagnosis. Neurocomputing 335, 327–335.

InvenSense, T.D.K., 2016. MPU-9250, nine-axis (gyro+accelerometer+compass) MEMS MotionTracking™ device. https://invensense.tdk.com/download-pdf/mpu-9250-datasheet/.

Kantue, Paulin, Pedro, Jimoh Olarewaju, 2020. Integrated fault detection and diagnosis of an unmanned aerial vehicle using time difference of arrival. In: 2020 24th International Conference on System Theory, Control and Computing. ICSTCC, pp. 336–342.

Keipour, Azarakhsh, Mousaei, Mohammadreza, Scherer, Sebastian, 2019. Automatic real-time anomaly detection for autonomous aerial vehicles. In: 2019 International Conference on Robotics and Automation. ICRA, IEEE.

Khalastchi, Eliahu, Kaminka, Gal A., Kalech, Meir, Lin, Raz, 2011. Online anomaly detection in unmanned vehicles. In: The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1. pp. 115–122.

Liang, Shaojun, Zhang, Shirong, Huang, Yuping, Zheng, Xing, Cheng, Jian, Wu, Sisi, 2022. Data-driven fault diagnosis of FW-UAVs with consideration of multiple operation conditions. ISA Trans. 126, 472–485.

Meng, Xiangrui, 2015. Simpler online updates for arbitrary-order central moments. arXiv preprint arXiv:1510.04923.

Ortiz-Torres, Gerardo, Castillo, Pedro, Sorcia-Vázquez, Felipe DJ, Rumbo-Morales, Jesse Y, Brizuela-Mendoza, Jorge A, De La Cruz-Soto, Javier, Martínez-García, Mario, 2020. Fault estimation and fault tolerant control strategies applied to VTOL aerial vehicles with soft and aggressive actuator faults. IEEE Access 8, 10649–10661.

Park, Jongho, Jung, Yeondeuk, Kim, Jong-Han, 2022. Multiclass classification fault diagnosis of multirotor UAVs utilizing a deep neural network. Int. J. Control Autom. Syst. 20 (4), 1316–1326.

Pebay, Philippe Pierre, Terriberry, Timothy, Kolla, Hemanth, Bennett, Janine Camille, 2014. Formulas for the computation of higher-order central moments. Tech. rept., Sandia National Lab.(SNL-CA), Livermore, CA (United States.

Pébay, Philippe, Terriberry, Timothy B, Kolla, Hemanth, Bennett, Janine, 2016. Numerically stable, scalable formulas for parallel and online computation of higher-order multivariate central moments with arbitrary weights. Comput. Statist. 31, 1305–1325.

Peng, Z.K., Peter, W. Tse, Chu, F.L., 2005. An improved Hilbert–Huang transform and its application in vibration signal analysis. J. Sound Vib. 286 (1–2), 187–205.

Shraim, Hassan, Awada, Ali, Youness, Rafic, 2018. A survey on quadrotors: Configurations, modeling and identification, control, collision avoidance, fault diagnosis and tolerant control. IEEE Aerosp. Electron. Syst. Mag. 33 (7), 14–33.

Simlinger, Benedict, Ducard, Guillaume, 2019. Vision-based gyroscope fault detection for UAVs. In: 2019 IEEE Sensors Applications Symposium. SAS, pp. 1–6.

Sun, Rui, Cheng, Qi, Wang, Guanyu, Ochieng, Washington Yotto, 2017. A novel online data-driven algorithm for detecting UAV navigation sensor faults. Sensors 17 (10).

Tan, Lynn Kai Lin, Lim, Beng Chong, Park, Guihyun, Low, Kin Huat, Yeo, Victor Chuan Seng, 2021. Public acceptance of drone applications in a highly urbanized environment. Technol. Soc. 64, 101462.

Thomas, Jibin B, Chaudhari, Saurabh G, Shihabudheen, KV, Verma, Nishchal K, 2023. CNN based transformer model for fault detection in power system networks. IEEE Trans. Instrum. Meas..

Wang, Benkuan, Peng, Xiyuan, Jiang, Min, Liu, Datong, 2020a. Real-time fault detection for UAV based on model acceleration engine. IEEE Trans. Instrum. Meas. 69 (12), 9505–9516.

Wang, Ban, Shen, Yanyan, Zhang, Youmin, 2020b. Active fault-tolerant control for a quadrotor helicopter against actuator faults and model uncertainties. Aerosp. Sci. Technol. 99, 105745.

Wang, Tao, Zhang, Le, Wang, Xuefei, 2023. Fault detection for motor drive control system of industrial robots using CNN-LSTM-based observers. CES Trans. Electr. Mach. Syst..

Wild, Graham, Murray, John, Baxter, Glenn, 2016. Exploring civil drone accidents and incidents to help prevent potential air disasters. Aerospace 3 (3), 22.

Wu, T.Y., Lei, K.W., 2019. Prediction of surface roughness in milling process using vibration signal analysis and artificial neural network. Int. J. Adv. Manuf. Technol. 102 (1), 305–314.

Xian, Bin, Hao, Wei, 2019. Nonlinear robust fault-tolerant control of the Tilt Trirotor UAV under rear Servo's stuck fault: Theory and experiments. IEEE Trans. Ind. Inform. 15 (4), 2158–2166.

Zhang, He, Gao, Qi, Pan, Feng, 2020. An online fault diagnosis method for actuators of quadrotor UAV with novel configuration based on IMM. In: 2020 Chinese Automation Congress. CAC, pp. 618–623.

Zhang, Xiaomin, Zhao, Zhiyao, Wang, Zhaoyang, Wang, Xiaoyi, 2021. Fault detection and identification method for quadcopter based on airframe vibration signals. Sensors 21 (2), 581.